



HTML 5



Agenda

1

Introduction to HTML5

2

HTML 5 Structural Elements

3

Creating Form Input and Validation

4

Graphics and Multimedia Elements

5

HTML Using Client-Side Storage Web Storage

Introduction to HTML5



Objectives

At the end of this sub-module you will know:

- What is HTML 5?
- Rules for HTML 5
- Browser support
- Features of HTML 5
- New elements in HTML 5
- New markup elements

What Is HTML 5

- The new version of HTML
- A new and emerging set of web standards and specifications
- HTML5 is HTML + CSS3 + JavaScript APIs



What is HTML 5 (Contd.).

- HTML5 is the next generation of HTML
- It will be the new standard for HTML, XHTML, and HTML DOM
- HTML5 work is still in progress and most modern browsers have some HTML5 support
- It is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG)
- WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0
- In 2006, they decided to collaborate and create a new version of HTML

What is HTML 5 (Contd.).

- It is a language for structuring and presenting content for the WWW
- It is a core technology of the Internet originally proposed by Opera Software
- The core aim of HTML5 is
 - Is to improve the language with support for multimedia
 - Easily read by humans
 - Understood by computers and devices
- The HTML5 working group includes Apple, Google, AOL, IBM, Microsoft, Mozilla, Nokia, Opera, and many hundreds of other vendors.

Rules for HTML 5

- New features should be based on HTML, CSS, DOM, and JavaScript
- Usage of external plug-ins (like Flash) should be reduced
- Strict parsing rules are introduced to handle any errors.
- More markup to replace scripting
- Device independence should be there

HTML5 and the Open Web Platform

- The Open Web Platform (OWP) is a collection of web technologies
- It is developed by W3C and other web standardization bodies
- HTML5 is part of the OWP

New Features of HTML5

- New canvas element is available for drawing
- New video and audio elements are available for media playback
- Better support for local offline storage
- New content specific elements, like article, mark, progress, meter
- New form controls, like calendar, date, time, email, url, search

The New Elements in HTML5

- The **canvas** element
- The **video** and **audio** elements
- New content-specific elements:
 - **article**, **footer**, **header**, **nav**, **section**, and more
- New form controls:
 - **calendar**, **date**, **time**, **email**, **url**, **search**, and more

The Doctype Element

- A declaration of document type
- Must be provided by any HTML document
- Defines the rendering mode of the browser
- Was created to enable HTML parsing and validation

```
<!DOCTYPE html>  
<html>
```

Browser Support

- No browsers have full HTML5 support as HTML 5 is yet not an official standard
- But all major browsers (Chrome, Safari, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions



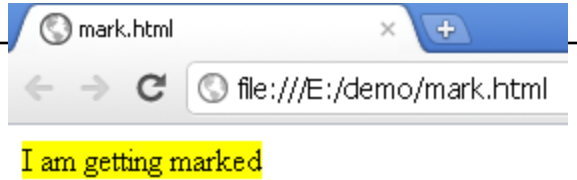
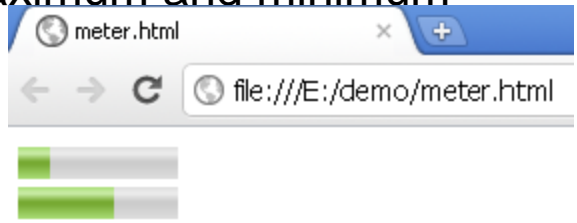
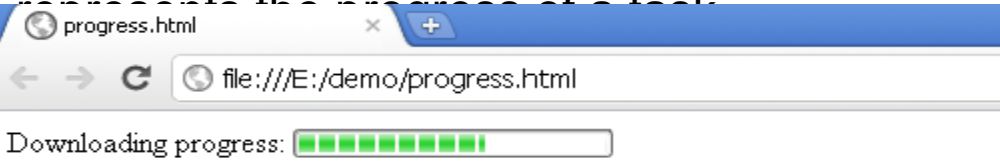
- Internet Explorer 9, Firefox, Opera, Chrome, and Safari support the `<video>` element.

Note: Internet Explorer 8 and earlier versions, do not support the `<video>` element.

New Elements in HTML5

- HTML5 includes new elements for better structure, drawing, media content and better form handling
- New media elements like `<audio>`, `<video>`, `<source>`
- Canvas element `<canvas>` uses JavaScript to make drawing on a web page
- The Input elements type attribute has many new values, for better input control before sending it to the server

Some new markup elements

| | |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <article> | Used to specify a news-article, blog post, forum post, or other articles which can be distributed independently from the rest of the site |
| <mark> | For text that should be highlighted  |
| <meter> | For a measurement, used only if the maximum and minimum values are known  |
| <progress> | The <progress> tag represents the progress of a task  Note: The progress element is currently supported in Firefox, Opera, and Chrome |

Summary

- At the end of this sub module:
 - Understand HTML 5
 - Different browser support for HTML 5
 - Different HTML 5 elements

HTML 5 Structural Elements



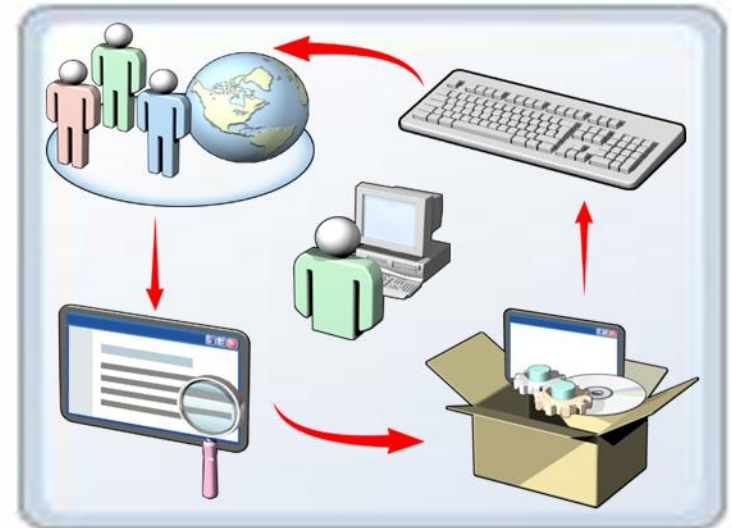
Objectives

- Explain the rationale behind the new HTML5 elements:
- Describe where and how to use the new HTML5 structural elements
- Describe what are the new semantic block-level elements
- Describe what are the new inline semantic elements
- Describe what are the new interactive elements

New HTML 5 – Rationale Behind

New HTML5 elements enable:

- Accessibility
- Searchability
- Internationalization
- Interoperability



The New Structural Elements

- New structural HTML5 elements:
 - **Article**
 - **Footer**
 - **Header**
 - **Hgroup**
 - **Section**
 - **Address**
 - **Nav**
 - **Aside**
- No need for **div** elements all over the webpage

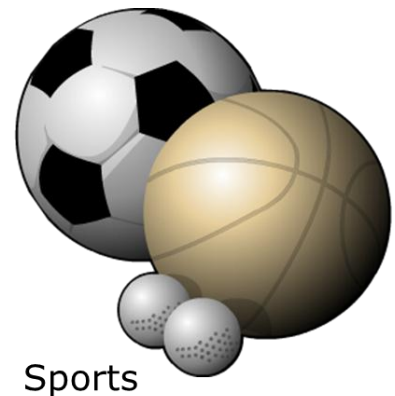
```
<body>
  <header>
    <nav>
      <ul>
        <li>Navigation</li>
      </ul>
    </nav>
  </header>
  <section>
    <article>
      <p>Article content</p>
    </article>
  </section>
  <footer>
    <p>Copyright</p>
  </footer>
</body>
```

Semantic Block-Level Elements

Semantic Block-Level Elements

- Block elements that have a specific semantic meaning
- Include the figure element

```
<figure>  
    
  <figcaption>Sports</figcaption>  
</figure>
```



Sports

Inline Semantic Elements

- Inline elements affect their content.
- Examples:
 - **Mark**
 - **Meter**
 - **Progress**
 - **Output**
 - **Time**

This is `<mark>`marked`</mark>` text

`<meter value="70" min="0" max="100" low="54" high="100" optimum="100">`70`</meter>`

Your download is `<progress max="100" value="60">`60%`</progress>` complete

Output: `<input type="text" value="1" />` + `<input type="text" value="3" />` =
`<output>`4`</output>`

The time is now `<time datetime="2011-09-28T18:36">`6:36 P.M. on November 28th`</time>`

This is **marked** text

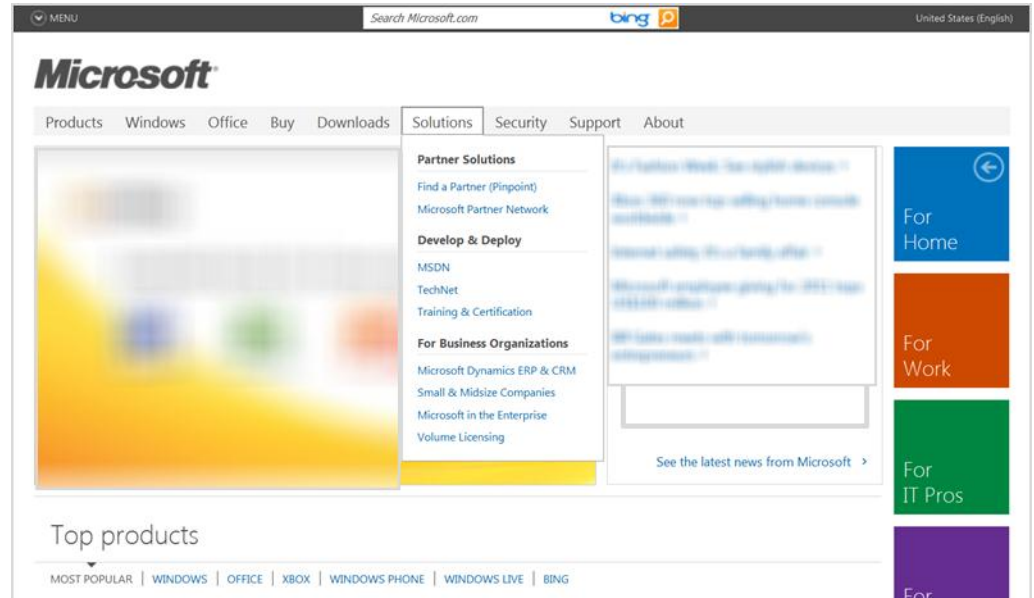
The time is now 6:36 P.M. on November 28th

Interactive Elements

- Interactive elements include:

- **Details**
- **Summary**
- **Menu**
- **Command**

```
<details>  
  <summary>The summary</summary>  
  <p>Some details</p>  
</details>
```



Summary

In the module, you learned about:

- The new HTML5 elements and the rationale behind the creation of these elements.
- How and when to use them.
- The use of navigation and menus.

Creating Form Input and Validation



Objectives

At the end of this sub-module you will know:

- HTML 5 new Input types
- HTML 5 new form elements
- HTML 5 new form attributes
- Using Browser Detection, Feature Detection, and Modernizr

Challenges with HTML 4.0 Forms

- The HTML 4 form fields are not validated
- The validation to the input fields can be done by
 - Writing JavaScript code in the Client side Code
 - Or the validation needs to be done at the Server side
- Almost every web page has some kind of form with search, email, sign-up etc
- It would be great if browsers had built in validation for some of the common data types we collect

HTML – New Input types

- HTML 5 has lot of new input types
- These new features allow for better input control and validation
 - email
 - url
 - number
 - range
 - Date pickers (date, month, week, time, datetime-local)
 - search
 - color

Input type - email

- It represents a control for editing a list of e-mail addresses given in the element's value
- HTML email address validation will only work in browsers which have support for it

```
<input type="email" placeholder="Please enter your email" required multiple="multiple">
```

- required - specifies that the element is a required part of form submission
- placeholder - a short phrase to help the user when entering data into the control
- multiple - specifies that the element allows multiple values

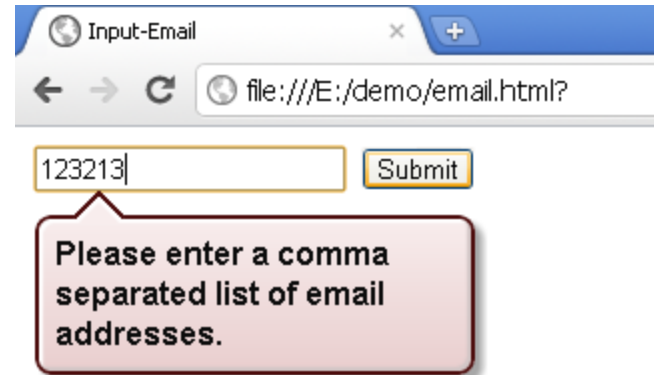
Input type – email (Contd.).

Field is empty



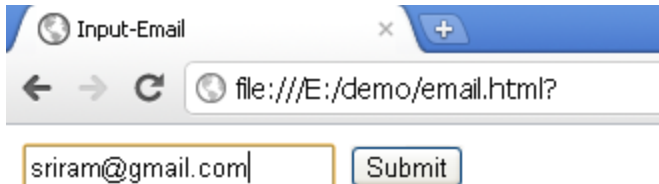
A screenshot of a web browser window titled "Input-Email". The address bar shows "file:///E:/demo/email.html". Below the address bar is an empty text input field and a "Submit" button. A red speech bubble with the text "Please fill out this field." points to the empty input field.

The value should be an email



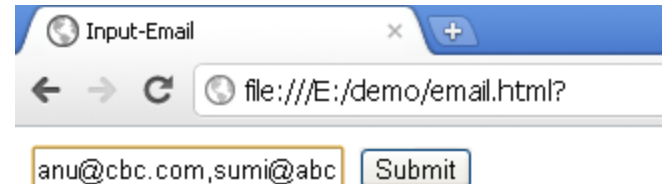
A screenshot of a web browser window titled "Input-Email". The address bar shows "file:///E:/demo/email.html?". Below the address bar is a text input field containing "123213" and a "Submit" button. A red speech bubble with the text "Please enter a comma separated list of email addresses." points to the input field.

Valid email – with a single email



A screenshot of a web browser window titled "Input-Email". The address bar shows "file:///E:/demo/email.html?". Below the address bar is a text input field containing "sriram@gmail.com" and a "Submit" button.

Valid-More than 1 email



A screenshot of a web browser window titled "Input-Email". The address bar shows "file:///E:/demo/email.html?". Below the address bar is a text input field containing "anu@cbc.com,sumi@abc" and a "Submit" button.

Input type - url

```
<input type="url" name="URL">
```

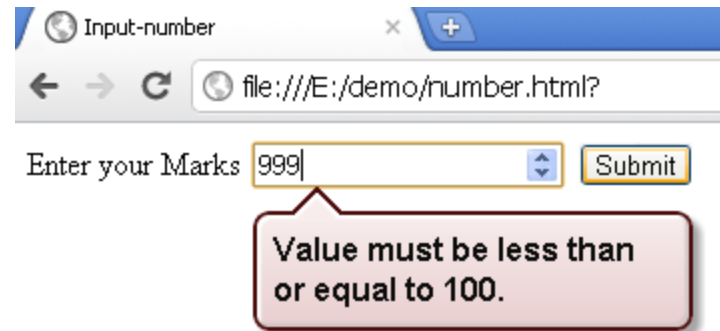
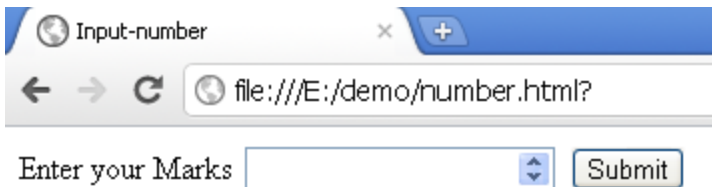
- It is used for input fields that should contain a URL address
- The value of the url field is automatically validated when the form is submitted



Input type - number

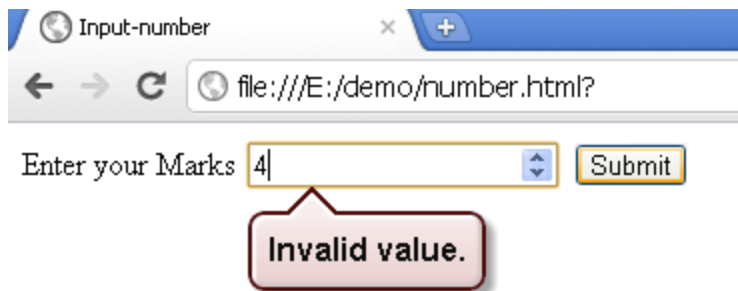
```
<input type="number" min="0" max="100">
```

- Used to take a number as input



```
<input type="number" min="0 " max="100" step="3">
```

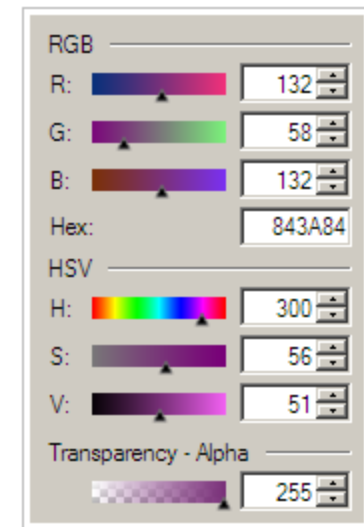
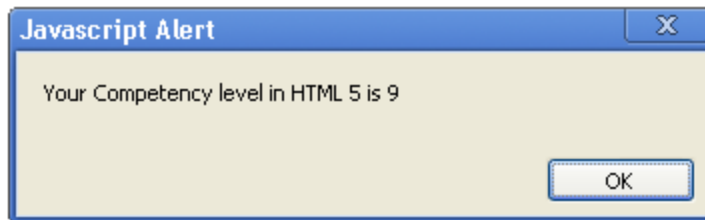
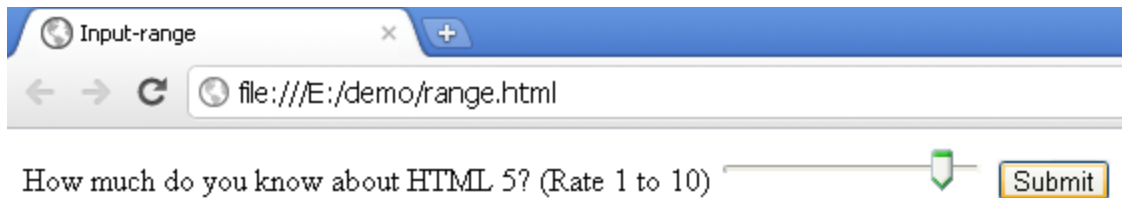
- step - specifies legal number intervals (if step="3", legal numbers could be -3,0,3,6, etc)



Input type - range

- Slider control is a very useful user interface to set a number within a range

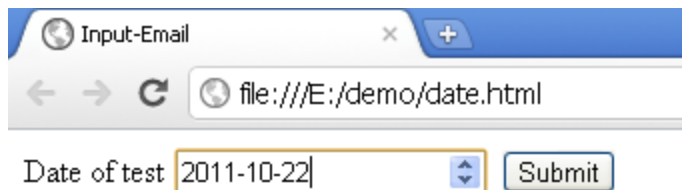
```
<input type="range" name="Range" min="1" max="10" required>
```



Input type – Date pickers - date

- Date and time field can be easily found in many web forms
- Typical applications are like ticket booking, appointment booking etc
- HTML5 the web browser ensures user can only enter a valid date time string into the input textbox

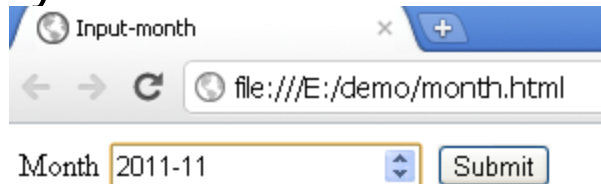
```
<input type="date" name="set_date" />
```



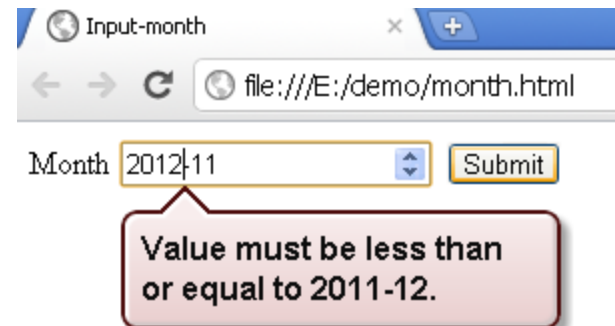
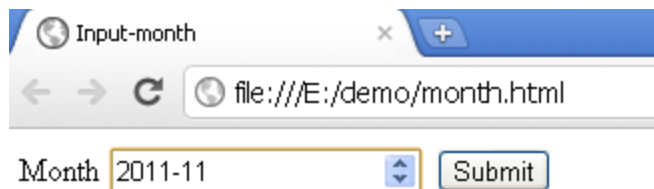
Input type – Date pickers – date (Contd.).

```
<input type="month" name="month" />
```

- Creates a date input control for specifying a particular month in a year



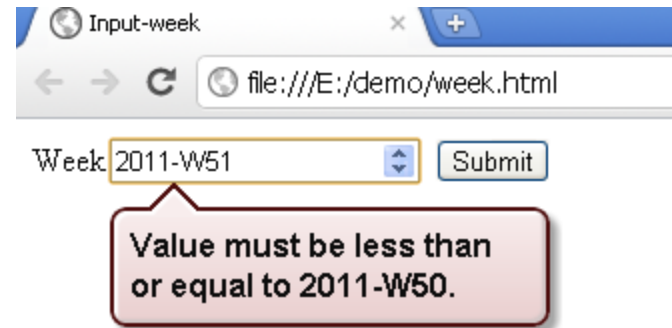
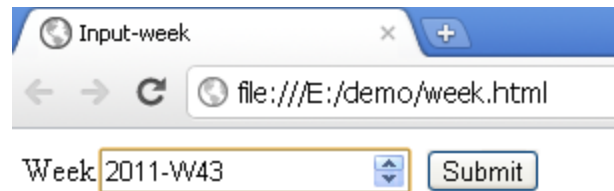
```
<input type="month" name="month" min="2011-1" max="2011-12"/>
```



Input type – Date pickers – week

```
<input type="week" name="week" min="2011-W15" max="2011-W50"/>
```

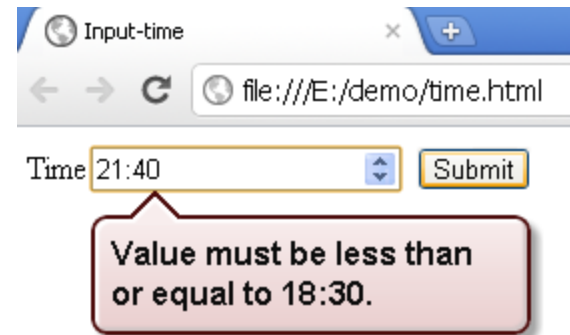
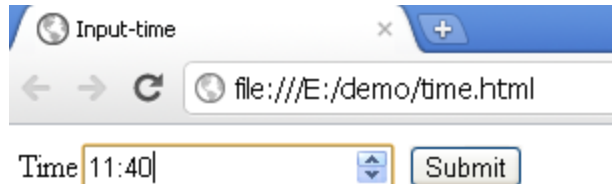
- Allows entry and validation of a week number



Input type – Date pickers – time

```
<input type="time" name="Time" min="11:40" max="18:30">
```

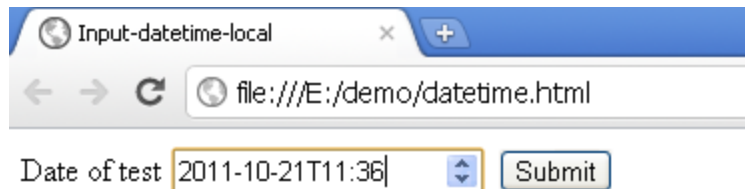
- Allows entry and validation of a valid time



Input type – Date pickers – datetime-local

```
<input type="datetime-local" name="set_date" />
```

- It creates a combined date/time input field
- The value is an ISO formatted date and time



Input type - Search

```
<input type="search" name="Search">
```

- It is used for search fields, like a site search, or Google search
- The difference between search and text type is only stylistic
- It takes the operating system's default rounded-corners style for search
- Currently Google Chrome does not support this



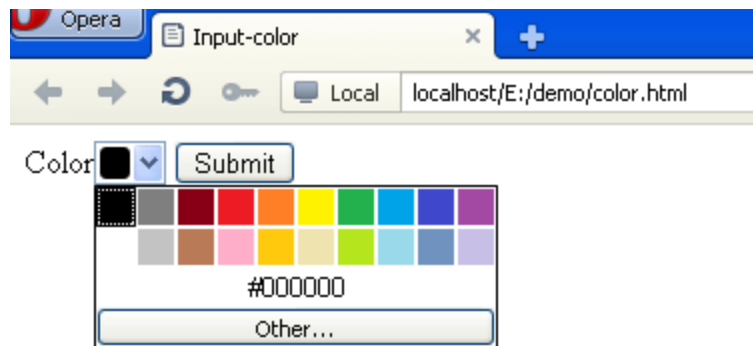
simplesearchinput

Input type - Color

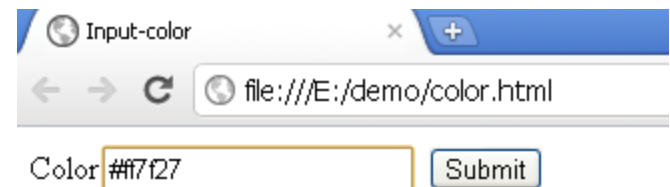
- Color `<input type="color" name="Color" required>`
- Used to create a color control for selecting a color value
- The Opera browser will allow you to select a color from a color picker

Color

In Opera 11.52



In Google chrome 14.0



HTML 5 Form Elements

- New Form elements added to HTML 5 are
 - `<datalist>` - A list of options for input values
 - `<output>` - For different types of output, such as output written by a script

HTML Form Attributes

- New Form Attributes
 - autocomplete
 - novalidate
- New Input Attributes
 - autocomplete
 - autofocus
 - form
 - formoverrides
 - height and width
 - list
 - min,max and step
 - multiple
 - pattern (regexp)
 - placeholder
 - required

Autocomplete attributes

- The **autocomplete** attribute instructs the browser to remember field values
- It work with <form> and the following input types
 - text, search, url, telephone, email, password, date pickers, range and color
- It can have two values on and off
- When the user starts to type in an auto complete field the browser should display options to fill in the field

Enabling Auto-Completion for a Form Field

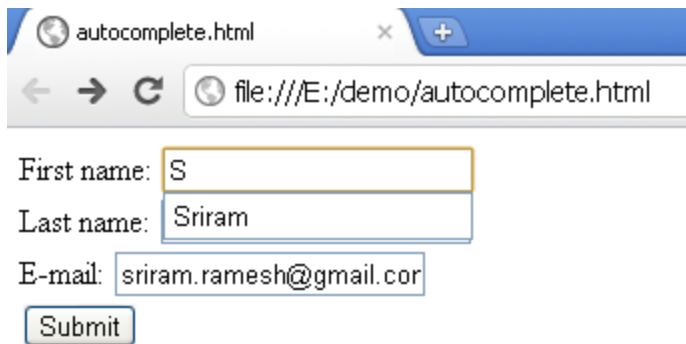
```
<input type="text" name="Username" autocomplete="on" />
```

Disabling Auto-Completion for a Sensitive Form Field

```
<input type="text" name="CreditCardNumber" autocomplete="off" />
```

Autocomplete attribute (Contd.).

```
<form action="/webapp/demo1" method="get" autocomplete="on">  
First name: <input type="text" name="fname" /><br />  
Last name: <input type="text" name="lname" /><br />  
E-mail: <input type="email" name="email" autocomplete="off" /><br />  
<input type="submit" />  
</form>
```

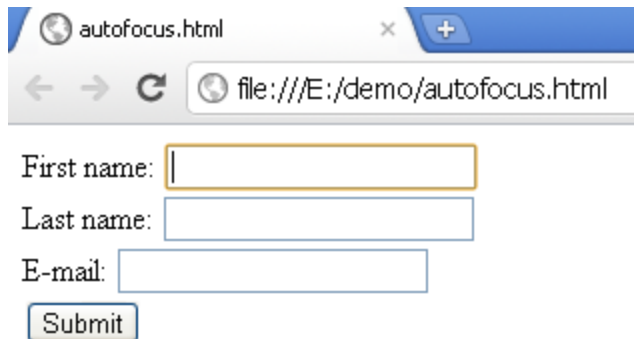


The screenshot shows a web browser window with the title 'autocomplete.html'. The address bar displays the file path 'file:///E:/demo/autocomplete.html'. The form contains three input fields: 'First name' with the value 'S', 'Last name' with the value 'Sriram', and 'E-mail' with the value 'sriram.ramesh@gmail.com'. A 'Submit' button is located below the email field. The 'autocomplete' attribute is set to 'on' for the form, which is evident from the pre-filled values in the input fields.

autofocus attribute

- It specifies that a field should automatically get focus when a page is loaded
- This attribute works with all `<input>` types

First name: `<input type="text" name="fname" autofocus="autofocus" />`

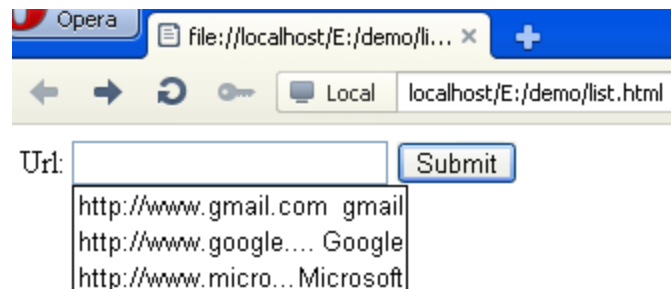


The screenshot shows a web browser window with the title 'autofocus.html'. The address bar displays the file path 'file:///E:/demo/autofocus.html'. The form contains three text input fields labeled 'First name:', 'Last name:', and 'E-mail:'. The 'First name:' field is highlighted with an orange border, indicating it has the autofocus attribute. Below the 'E-mail:' field is a 'Submit' button.

list attribute

- The list attribute specifies a datalist for an input field
- A datalist is a list of options for an input field
- The list attribute works with the following <input> types
 - text, search, url, telephone, email, date pickers, number, range, and color

```
Url: <input type="text" list="url_list" name="url" />
<datalist id="url_list">
  <option label="gmail" value="http://www.gmail.com" />
  <option label="Google" value="http://www.google.com" />
  <option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```



list Attribute and the datalist Element

- The **list** attribute provides another mechanism for auto-completion.
- Whereas the **autocomplete** attribute instructs the browser to suggest values based on historical entries that the browser itself manages, the **list** attribute gives the web developer more control over the suggestions.
- A list of pre-defined values can be specified in a **datalist** element somewhere on the page, and referenced by any text field.
- The browser then uses the pre-defined values as suggestions when the user fills the text field.
- To specify the suggestions made available by a **datalist** element, use **option** elements.
- The value of the **list** attribute is the ID of the **datalist** element. A **datalist** element contains **option** elements representing its values.

Usability Attributes

```
<input type="text" placeholder="Enter your full name"
name="FullName" />
<input type="text" name="FullName" autofocus="autofocus" />
<input type="text" name="Username" autocomplete="on" />
```

The diagram shows two examples of text input fields. The first, labeled "Textbox with Watermark", is a simple rectangular box containing the text "Please enter your name". A red arrow points to it from the label below. The second, labeled "Textbox with Autocomplete", shows a form with two fields. The first field is labeled "First name:" and contains the letter "J". The second field is labeled "E-mail:" and contains the text "John". A blue highlight is visible under the text "John" in the second field. A red arrow points to this highlight from the label below. A "Submit" button is located below the "E-mail:" field.

Please enter your name

Textbox with Watermark

First name: J

E-mail: John

Submit

Textbox with Autocomplete

form attribute

- In an HTML page there can be many number of forms but only one form can be submitted
- HTML 5 allows an element to use the form attribute and specify the form to which it belongs to

```
<form action="http://localhost:10000/webApp/default.aspx" method="get" id="user_form">
```

```
    First name:<input type="text" name="fname" />
```

```
    <input type="submit" />
```

```
</form>
```

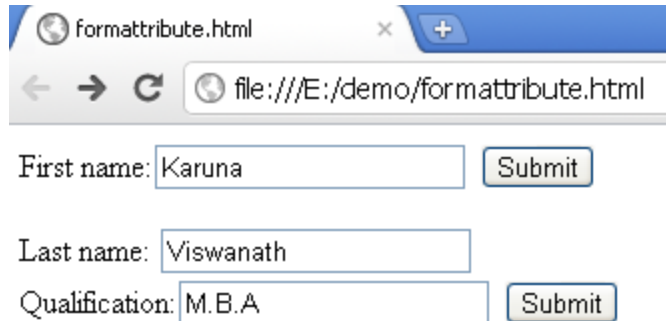
```
Last name: <input type="text" name="lname" form="user_form" />
```

```
<form action="http://localhost:10000/webApp/default1.aspx" method="get" id="user_form2">
```

```
    Qualification:<input type="text" name="qualification" form="user_form"/>
```

```
    <input type="submit" /></form>
```

form attribute (Contd.).



formattribute.html

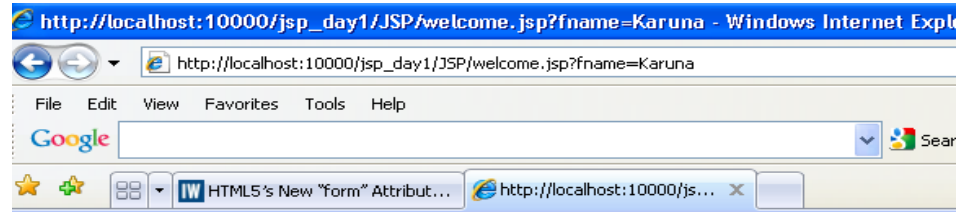
file:///E:/demo/formattribute.html

First name:

Last name:

Qualification:

It works like this in an unsupported browser



Welcome Karuna null
You have done your null

Output in a browser with form attribute support



Welcome Karuna Viswanath
You have done your M.B.A

formoverrides attribute

- It allows you to override some of the attributes set for the form element
- The various form override attributes are:
 - formaction – It overrides the form action attribute
 - formmethod – It overrides the form method attribute
 - formnovalidate – It overrides the form novalidate attribute
 - formtarget – It overrides the form target attribute
- The form override attributes works with the following `<input>` types
submit and image

formation attribute

- It overrides the form action attribute

```
<form action="/WebApp/user.aspx" method="get">  
Email: <input type="email" name="userid">  
<input type="submit" value="user">  
<input type="submit" formaction="/WebApp/admin.aspx"  
      value="admin">  
</form>
```

- If the user clicks the user button it submits the form to user.aspx
- If the user clicks the admin button it submits the forms to admin.aspx

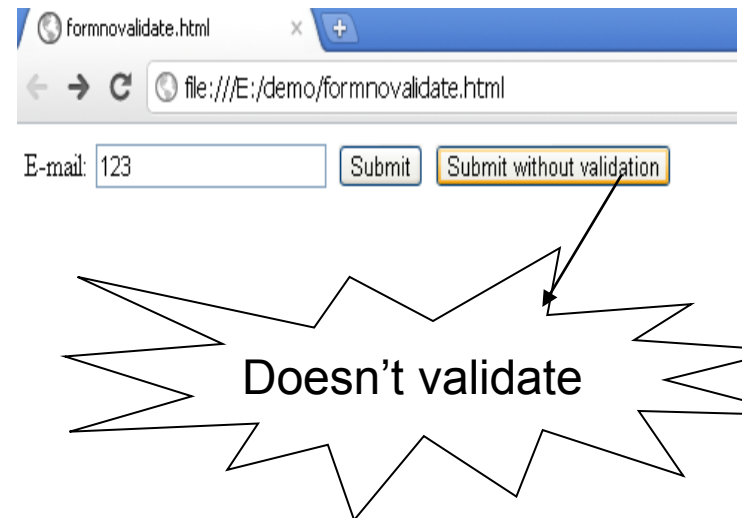
formmethod attribute

- Helps to override the form method attribute (get / post)

```
<form action="one.aspx" method="get">
FName: <input type="text" name="firstname" /><br />
LName: <input type="text" name="lastname" /><br />
<input type="submit" value="Submit" />
<input type="submit" formmethod="post" formaction="two.aspx"
value="Submit using POST" />
</form>
```

formnovalidate attribute

```
<form action="demo.aspx" method="get">  
E-mail: <input type="email" name="userid">  
<input type="submit">  
<input type="submit" formnovalidate="true" value="Submit without  
validation">  
</form>
```



formtarget attribute

- Used to override the target attribute of the form element

```
<input formtarget="_blank|_self|_parent|_top|framename" />
```

| | |
|-----------|--------------------------------------------------------------------|
| _blank | Submits the form to a document in a new window or tab |
| _self | Submits the form to a document in the same frame (this is default) |
| _parent | Submits the form to a document in the parent frame |
| _top | Submits the form to a document in the full body of the window |
| framename | Submits the form to a document in a named iframe |

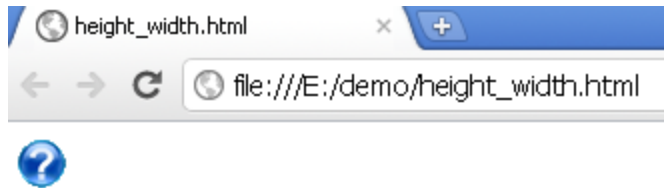
formtarget attribute (Contd.).

```
<form action="demo1.aspx" method="get">  
  First name: <input type="text" name="firstname" /><br />  
  Last name: <input type="text" name="lastname" /><br />  
  <input type="submit" value="Submit" />  
  <input type="submit" formtarget="_blank" value="Submit to a new  
    window/tab" />  
</form>
```

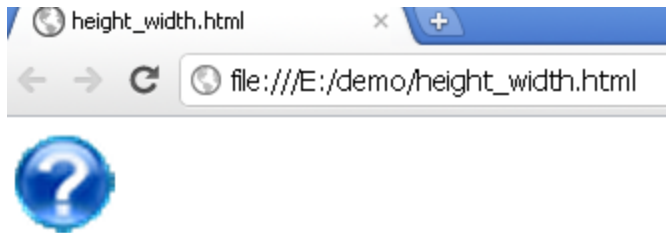

height and width attribute

- height - specifies the height of the image
- width - specifies the width of the image

```
<input type="image" src="q-mark.gif" />
```



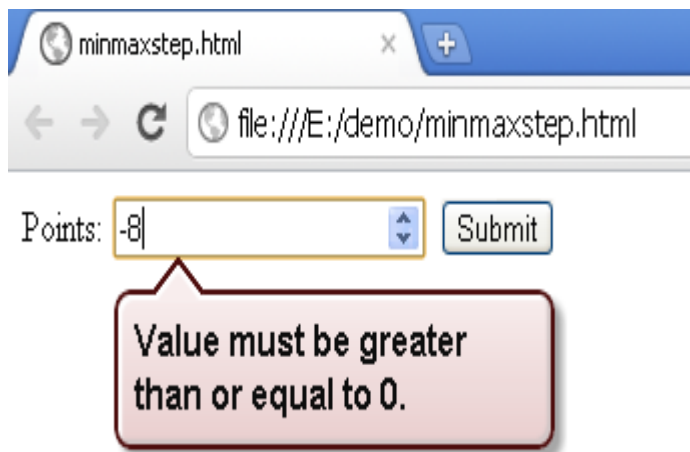
```
<input type="image" src="q-mark.gif" height="50" width="50" />
```



min, max and step attribute

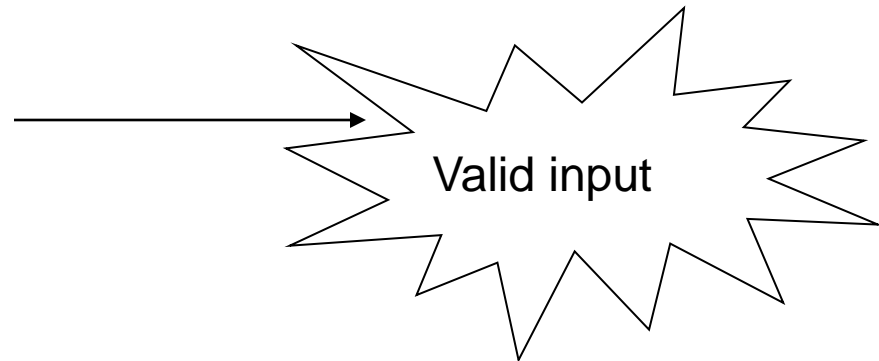
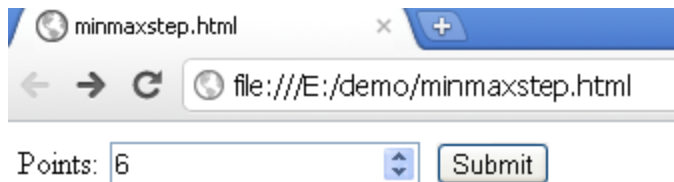
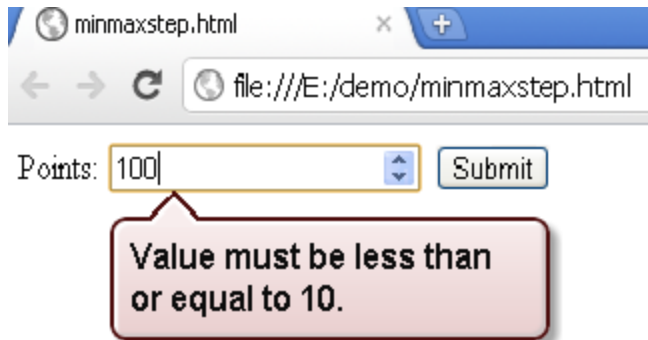
- max – used to specify the maximum value allowed for the input field
- min - used to specify the minimum value allowed for the input field
- step - specifies the legal number intervals for the input field (if step="4", legal numbers could be -4,0,4,8, etc)
- The min, max, and step attributes works with the following <input> types
 - date pickers, number, and range

Points: `<input type="number" name="pts" min="0" max="10" step="3"/>`



min,max and step attribute (Contd.).

Points: `<input type="number" name="pts" min="0" max="10" step="3"/>`



multiple attribute

- The multiple attribute is used to indicate if the user can be allowed to specify more than one value

Email - To: `<input type="email" name="to" />`



multiple.html

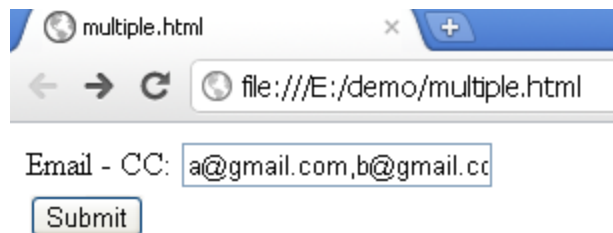
file:///E:/demo/multiple.html

Email - To: @gmail.com,b@gmail.com

Submit

Please enter an email address.

Email - CC: `<input type="email" name="cc" multiple />`



multiple.html

file:///E:/demo/multiple.html

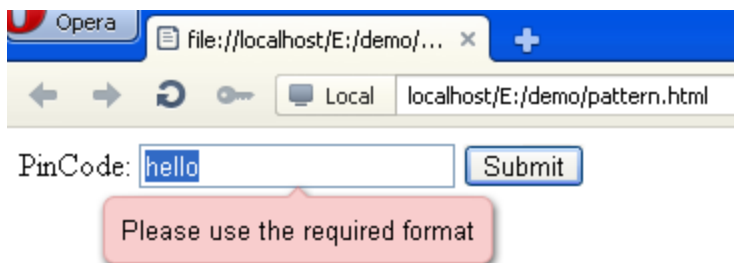
Email - CC: a@gmail.com,b@gmail.cc

Submit

pattern attribute

- The pattern attribute is used to specify a pattern to validate an input field
- The pattern is a regular expression
- The pattern attribute works with the following <input> types
 - text, search, url, telephone, email, and password

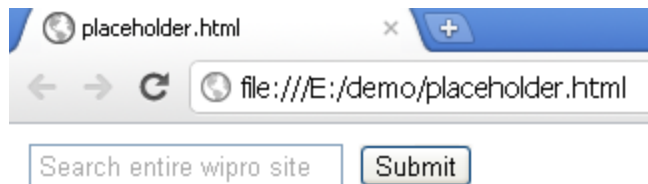
```
<input type="text" pattern="[0-9]{6}" name="pincode" placeholder="A  
pincode is a 6 digit number"/>
```



placeholder attribute

- It provides description about the expected value of an input field
- It works with the following <input> types
 - text, search, url, telephone, email, and password
- The hint is displayed in the input field when it is empty, and disappears when the field gets focus

```
<input type="search" name="usr_search" placeholder="Search entire wipro site" />
```



required attribute

- The required attribute specifies that an input field must be filled out before submitting
- It works with the following `<input>` types
 - text, search, url, telephone, email, password, date pickers, number, checkbox, radio, and file

Name: `<input type="text" name="user_name" required="required" />`



Cross-Browser Interoperability

- One of the main problems with earlier versions of HTML was cross-browser interoperability
- Developers strive for code that will work in all browsers
 - This is impossible because of legacy browsers and the difference between browser engines

Browser Detection

- A technique to check which browser is running the code
- Can make dozens of assumptions based on this single check
- Should be avoided most of the time
- Should only be used when applying code to a specific legacy browser of a known version

```
<!--[if IE 7]><div>rendered in Internet Explorer 7 only</div><![endif]-->
```

Feature Detection

- A technique to check the availability of a feature before it is used
- Standard features should be looked for first
 - Browsers sometimes support both the standards and a legacy implementation
 - Looking for standard features first will ensure the use of standards if available
- Avoid making any assumptions about existing features

```
if (window.addEventListener) {  
    // HTML that will be rendered if addEventListener is available  
}  
else {  
    // HTML that will be rendered if addEventListener is not available  
}
```

Modernizr

- A small JavaScript library
- Detects availability of HTML5 and CSS3 specifications and whether features are natively implemented in the browser
- Uses feature detection under the hood

```
if (Modernizr.canvas) {  
    // HTML that will be rendered if the Canvas element is available  
}
```

Summary

- Describe the new HTML5 input types.
- Apply date pickers on webpages.
- Use the new text box types on webpages.
- Use the new interactive text box types on webpages.

Graphics and Multimedia Elements



Objectives

At the end of this sub-module you will learn:

- Use the canvas element and manipulate it by using JavaScript.
- Use the audio and video elements, and control them by using JavaScript.
- Add support for multiple audio and video codecs.

Canvas Basics

- Working with the Canvas
- Drawing Shapes
- Applying Styles
- Drawing Images
- Drawing Text
- Transformations

<canvas> tag

- It is used to draw graphics on a web page
- It is a rectangular area and every pixel of it can be controlled
- Used to draw simple diagrams, animations, charts and graphics
- Uses JavaScript to do the drawing
- Has several methods for drawing paths, boxes, circles, characters, and adding images.

Working with the Canvas

- The canvas element is a drawing surface
- The canvas element is lookless
- The “2d” context draws two-dimensional drawings on the canvas

```
<canvas id="canvas" width="300" height="150"></canvas>
```

- After the canvas is declared, the element can be retrieved by using Javascript

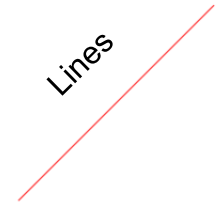
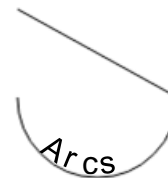
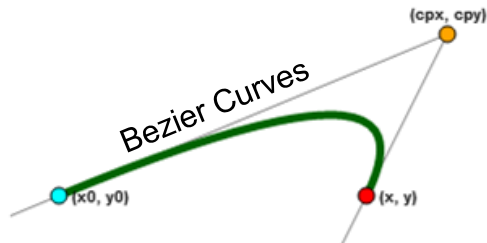
```
var canvas = document.getElementById("canvas");  
var context = canvas.getContext("2d");
```

The canvas is fun!
The canvas is fun!

Drawing Shapes

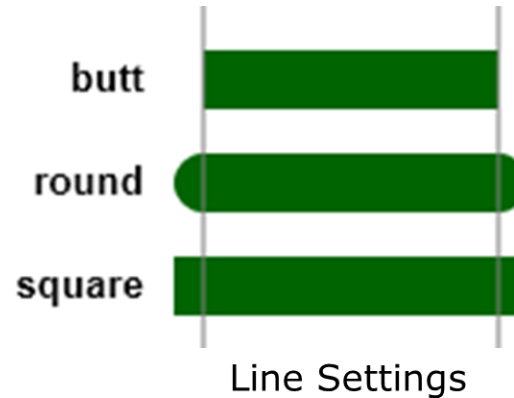
- Simple Shapes: Rectangles
- Complex Shapes: Paths
 - Straight Lines
 - Arcs
 - Bezier Curves
 - Rectangles
 - Clipping Regions

```
context.fillRect(x, y, width, height)
context.strokeRect(x, y, width, height)
context.clearRect(x, y, width, height)
```

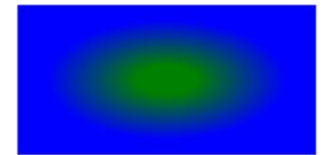


Applying Styles

- Colors
- Gradients
 - Linear Gradients
 - Radial Gradients
- Line Settings



Linear Gradient



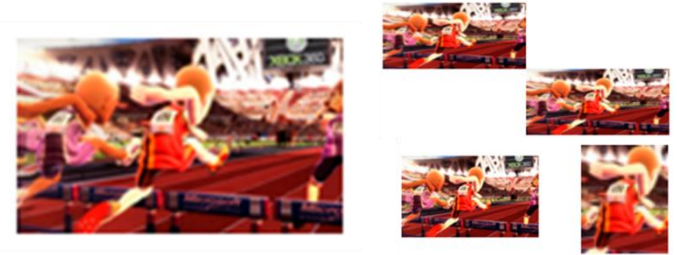
Radial Gradients

```
context.fillStyle = style  
context.strokeStyle = style
```

```
context.createLinearGradient(x0, y0, x1, y1)  
gradient.addColorStop(0.25, "red");  
gradient.addColorStop(0.5, "green");  
context.fillStyle = gradient;
```

Drawing Images

- Images can be drawn from various sources:
 - **img** elements
 - **video** elements
 - **canvas** elements
- The **drawImage** overloads can draw, scale, and slice images.
- Patterns of repeated images can be used as stroke and fill styles.



```
var image = document.createElement("img");
image.onload = function () {
    context.drawImage(image, 20, 20);
};
image.src = "image.png";
```

Drawing Text

- The **font** property is set by using a CSS-style declaration
- Text on the canvas uses many CSS paradigms
- Text can be aligned horizontally or vertically
- Multiline text is not supported
- Use the start and end alignments for multilingual support

```
context.fillText(text, x, y);  
context.strokeText(text, x, y);
```

Playing Multimedia

- The **video** and **audio** elements are interchangeable
- The media file is specified with the **src** attribute
- The **autoplay** and **loop** attributes control automatic playback
- The **preload** attribute recommends a buffering strategy
- The **volume** and **muted** attributes control the volume
- The **controls** attribute toggles the built-in media controls
- Add fallback content to support legacy browsers

```
<video src="video.m4v"></video>  
<audio src="audio.m4a"></audio>
```

Video-Specific Attributes

- The **video** element can be laid out easily by using CSS
- The **width** and **height** attributes specify the dimensions of the viewing area
- The video always maintains its innate aspect ratio
- The **poster** frame image is shown before the video is played

```
<video src="video.m4v" poster="first_frame.png"  
width="100px" height="100px"></video>
```

Alternative Sources

- The **source** element specifies alternate media resources
- Multiple **source** elements can be specified
- The browser uses only the first **source** element it supports
- Specifying the MIME type with the **type** attribute avoids downloading unsupported files
- The **media** attribute constrains the environment by using *media queries*

```
video>  
  <source src="video.m4v" " type="video/mp4" />  
</video>  
<audio>  
  <source src="audio.m4a" " type="video/mp4" />  
</audio>
```


Video and Audio Formats



| Format | IE | Firefox | Opera | Chrome | Safari |
|--------|-----|---------|-------|--------|--------|
| Ogg | No | Yes | Yes | Yes | No |
| MP3 | Yes | No | No | Yes | Yes |
| Wav | No | Yes | Yes | Yes | Yes |

Controlling Playback

- The **play** method starts playback
- The **pause** method pauses playback
- The **playbackRate** property changes the playback speed
- The **currentTime** property reports the current position and seeks
- The **duration** property reports the length of the video or audio
- The **buffered** property lists the currently downloaded data
- Many properties and events report status and progress

```
var video = document.getElementById("video");
video.play();
window.setTimeout(function () {
    video.pause();
}, 3000);
```

Integrating Video and the Canvas

- Video uses time, and the canvas does not
- Time loops are used to simulate time
- The **setInterval** and **setTimeout** methods drive the time loop
- The Video and Canvas APIs work well together

```
function frame() {  
    var context =  
document.getElementById("canvas").getContext("2d");  
    var video = document.getElementById("video");  
    context.drawImage(video, 0, 0);  
    window.setTimeout(frame, 100);  
}  
  
frame(); // calling the function directly triggers the  
first iteration
```

Summary

- Introduced some of the most exciting features in HTML5. The multimedia elements—**video** and **audio**
- Many accessibility features.
- Understanding APIs that programmers can use to control
- multimedia.

Introduction to JQuery & Syntax



Objectives

At the end of this module you will understand

- What is jQuery?
- jQuery Features
- How jQuery Works
- Downloading jQuery Library
- Using jQuery Library
- Basic jQuery Example

What is jQuery?

- jQuery is an open-source library of JavaScript functions
- jQuery project started in 2006 by John Resig
- jQuery helps you to easily & quickly write clean JavaScript
- The jQuery library contains many common functions & plug-ins that you can extend immediately. For example:
 - Applying styles on HTML DOM elements on-the-fly
 - Use AJAX out-of-the-box
 - Add date picker, auto-complete, progress bar, menu tabs to your forms
 - Make animations almost as good as Flash can

JQuery

- jQuery is a fast, lightweight JavaScript library that is CSS3 compliant and supports many browsers.
- The jQuery framework is extensible and very nicely handles DOM manipulations, CSS, AJAX, Events and Animations.
- JavaScript is a language whereas jQuery is a library written using JavaScript.

Why jQuery?

- Consider that we have to apply zebra-stripes on a table with many rows
- This can be achieved by either using CSS or JavaScript
- Using CSS has an immediate fall-out – Row Consistency

| | | |
|----|-------------------------------|-------------------|
| 1 | Lost In The Plot | The Dears |
| 2 | Poison | The Constantines |
| 3 | Plea From A Cat Named Virtute | The Weakerthans |
| 4 | Melissa Louise | Chixdiggit! |
| 5 | Living Room | Tegan And Sara |
| 6 | Speed | Bran Van 3000 |
| 7 | Fast Money Blessing | King Cobb Steelie |
| 8 | Sore | Buck 65 |
| 9 | Love Travel | Danko Jones |
| 10 | You Never Let Me Down | Furnaceface |

- A typical JavaScript function will require about a few dozen lines of code and could look something like this:

Zebra-stripping with JavaScript

```
var stripe = function() {  
    var tables = document.getElementsByTagName("table");  
    for(var x=0;x!=tables.length;x++){  
        var table = tables[x];  
        if (! table) { return; }  
        var tbodies = table.getElementsByTagName("tbody");  
        for (var h = 0; h < tbodies.length; h++) {  
            var even = true;  
            var trs = tbodies[h].getElementsByTagName("tr");  
            for (var i = 0; i < trs.length; i++) {  
                this.className += " ruled"; return false  
                if(even) trs[i].className += " even";  
                even = !even;  
            }  
        }  
    }  
}  
window.onload = stripe;
```

Zebra-striping with jQuery

```
$("#tr:nth-child(even)").css('background-color','coral');
```

What does jQuery do?

- **Crunches JavaScript**
 - reduces the number of lines of JavaScript you would have to write, call it shortcut JavaScript
- **Manipulate HTML**
 - allows you to select & manipulate HTML, popularly used to manipulate <form> elements
- **Manipulate CSS**
 - allows you to apply, remove or change CSS properties applied on HTML elements, on-the-fly
- **Out-of-the-box widgets**
 - includes many commonly needed widgets. For example, calendar popups, accordion, tab panels, etc
- **Animation**
 - apply animated effects like show, hide, slide on HTML elements. Also includes some very cool animation effects on images.

Downloading jQuery Library

- Download the latest release of the library from **jQuery.com**
- 2 versions are available – minified & uncompressed

- Alternatives to downloading:

- Google CDN:

```
<head>  
<script type="text/javascript"  
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>  
</head>
```

- Microsoft CDN:

```
<head>  
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery/jquery-  
1.4.2.min.js"></script>  
</head>
```

Basic jQuery Usage Example

```
<html>
<head>
<script type="text/javascript" src="myJsPath/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
</head>
```

Add the jQuery library file in the head section



```
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

How does jQuery work?

- jQuery works first by “Selecting” the HTML element(s) and then applying your “Action” on it
- To select and apply action, use the **jQuery** object
- The jQuery object can also be called using **\$**
- Example:

```
// Select all elements of class: "someClass" and  
// apply a red border to all of them  
jQuery(".someClass").css("border", "1px solid red");
```

```
// Exactly the same as above, but using $  
$(".someClass").css("border", "1px solid red");
```

jQuery Syntax - \$() Factory

- All types of selectors in jQuery start with \$()
- The syntax is **\$(selector).action()**
 - The \$ sign defines jQuery
 - A selector to “query or find” HTML elements
 - A jQuery action() to perform on the HTML elements
- Examples:
 - \$(this).hide → hides current element
 - \$("p").hide → hides all paragraphs
 - \$("p.test").hide → hides all paragraphs with class="test"
 - \$("#test").hide → hides the element with id="test"

The Document Ready Function

```
$(document).ready(function(){  
    // jquery function go here...  
});
```

- This method of writing function, prevents the jQuery function from running before the document is finished loading (is ready)
- Note: It is not mandatory to write jQuery statements within this construct

`$(document).ready()` function

- If you want an event to work on your page, you should call it inside the `$(document).ready()` function.
- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.
- The `$(document).ready()` function has some benefit for its own way to make its events to work.
- First no need for "behavioral" markup tag in the HTML.
- `$(document).ready()` helps your events to get load or fire before the window loads.
- Anything that you write inside its brackets is ready as soon as the DOM is registered by the browser.

How to write jQuery functions

- Simple function:

```
$(document).ready(function(){  
    $("p").hide();  
}); // end document.ready
```

- Nested function:

```
$(document).ready(function(){  
    $("button").click(function(){  
        var num1 = 10;  
        var num2 = 5;  
        var numSum = num1 + num2;  
        alert("The sum is " + numSum);  
    }); // end click  
}); // end document.ready
```

jQuery

jQuery Syntax

- The jQuery syntax is made for **selecting** HTML elements and perform some **action** on the element(s).
- Syntax **\$(selector).action()**
- A dollar sign to define jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

Summary

- At the end of this sub module
 - What is jQuery
 - jQuery features
 - Downloading jQuery library
 - Including jQuery in web pages
 - How jQuery works
 - The `$()` and `jQuery()` function
 - jQuery chaining
 - The `$(document).ready()` function
 - Writing jQuery functions

jQuery Selectors



Objective

- At the end of this sub module we will understand
 - Element Selectors
 - CSS Selectors
 - Attribute Selectors
 - Advanced Selectors

jQuery Selectors

- In HTML DOM, selectors allow you to manipulate DOM elements as a group or as a single node
- jQuery selectors allow you to select HTML elements (or groups of elements) by element name, attribute name or by content
- Types of jQuery selectors:
 - Element Selectors
 - CSS Selectors
 - Attribute Selectors

jQuery Element Selectors

- Element Selectors
 - jQuery uses the HTML tags to select HTML elements
 - Example:
 - `$("p")` → selects all `<p>` elements
 - `$("a")` → selects all `<a>` elements
 - `$("ul")` → selects all `` elements
 - `$("*")` → selects all elements within the document
 - `$("img, span")` → selects `` and `` elements

jQuery CSS Selectors

- CSS Selectors
 - jQuery uses CSS IDs & classnames to select elements
 - A CSS ID is preceded with a '#' & a classname with a '.'
 - Examples:
 - `$("#intro")` → selects the first element with id="intro"
 - `$("p.intro")` → selects all <p> elements with class="intro"
 - `$("a#myId.myClass")` → selects <a> with id="myId" & class="myClass"
 - `$("p a.specialClass")` → selects <a> with class="specialClass" declared within <p> elements

jQuery Attribute Selectors

- Attribute Selectors
 - jQuery uses XPath expressions to select elements with given attributes
 - To select, put the attribute with in []
 - Examples:
 - `$("[href]")` → selects all elements with href attributes
 - `$("input[type=text]")` → selects all <input> elements with a type of text

jQuery Advanced Selectors

- jQuery advanced selector uses the next generation of CSS supported by Mozilla Firefox, IE 7, Chrome, Safari, etc
- Examples:
 - `$('*')` → Selects all elements in the document
 - `$(this)` → Selects the current element
 - `$("p > *")` → Selects all elements that are children of `<p>` element
 - `$("li:not(.myclass)")` → Selects all elements matched by `` that do not have `class="myclass"`
 - `$("li > ul")` → Selects all elements matched by `` that are children of an element matched by ``
 - `$("code, em, strong")` → Selects all elements matched by `<code>` or `` or ``
 - `$("p:empty")` → Selects all elements matched by `<p>` that have no children
 - `$(":radio")` → Selects all radio buttons in the form
 - `$(":checked")` → Selects all checked boxes in the form
 - `$("[href!='#']")` → Selects all elements with href value NOT equal to `"#"`
 - `$("[href^='http://']")` → Selects all elements with href value that starts with `'http://'`
 - `$("[href$='.pdf']")` → Selects all elements with href value that ends with `' .pdf'`

jQuery Advanced Selectors

- `$("li:even")` → Selects all elements matched by `` that have an even index value
- `$("tr:odd")` → Selects all elements matched by `<tr>` that have an odd index value
- `$("li:first")` → Selects the first `` element
- `$("li:last")` → Selects the last `` element
- `$("li:eq(2)")` → Selects the third `` element. Index starts at 0
- `$("li:lt(2)")` → Selects all elements matched by `` element before the third one; in other words, the first two `` elements
- `$("li:gt(1)")` → Selects all elements matched by `` after the second one
- `$("li:first-child")` → Selects all elements matched by `` that are the first child of their parent
- `$("li:last-child")` → Selects all elements matched by `` that are the last child of their parent
- `$(":parent")` → Selects all elements that are the parent of another element, including text
- `$("li:contains(second)")` → Selects all elements matched by `` that contain the text second
- `$("li:visible")` → Selects all elements matched by `` that are visible
- `$("li:hidden")` → Selects all elements matched by `` that are hidden

* *The above examples are not limited to `` & also apply to other elements like `<p>`, ``, etc*

jQuery Selector Usage

```
<html>
<head>
<title>jQuery - Selectors</title>
<script type="text/javascript" src="jsLib/jquery-1.4.2.min.js"></script>
</script>
</head>

<body>
<p id="intro">This is just a test of JQuery.</p>
<ol>
  <li>Bangalore</li>
  <li>Hyderabad</li>
  <li>Kolkata</li>
  <li>Chennai</li>
  <li>Pune</li>
  <li>Bhubaneswar</li>
</ol>
<p>This is how it is defined </p>
```

jQuery Selector Usage (cont'd)

```
<script type="text/javascript">
$(document).ready(function(){
    $("p").css("background-color", "#ff9");
    $("a").css("background-color", "#f00");
    $("p#intro").css("border", "3px dashed #00f");
    $("li:first").css("background-color", "#66f");
    $("li:contains(jQ)").css("background-color", "#0f0");
});
</body></html>
```

Hands-On-Lab

- Create a HTML document with:
 - Add the jQuery library to your HTML document using `<script>`
 - `<h1>` header
 - `<p>` with some content in it
 - An ordered list `` with about 6 `` elements
 - Another `<p>` with some content in it and few of the phrases with `<a>` link on it
- The first and the last `` element should have the background color as red and green respectively.

Summary

- In this session, we learnt about:
 - Element Selectors
 - CSS Selectors
 - Attribute Selectors
 - Advanced Selectors

Mobile Application Introduction



Objective

- At the end of this sub module we will understand
 - Platforms , tools and technologies
 - Application Design considerations
 - Understand the .NET Compact Framework
 - Mobile Application Architecture

Introduction

- Mobile applications can now be developed to deliver any types of data, to any user, any place in the world!
- .NET Mobile is an extension to Microsoft ASP.NET and the Microsoft's .NET Framework.
- .NET Mobile is a set of server-side Forms Controls to build applications for wireless mobile devices.
- These controls produce different output for different devices by generating WML, HTML, or compact HTML.

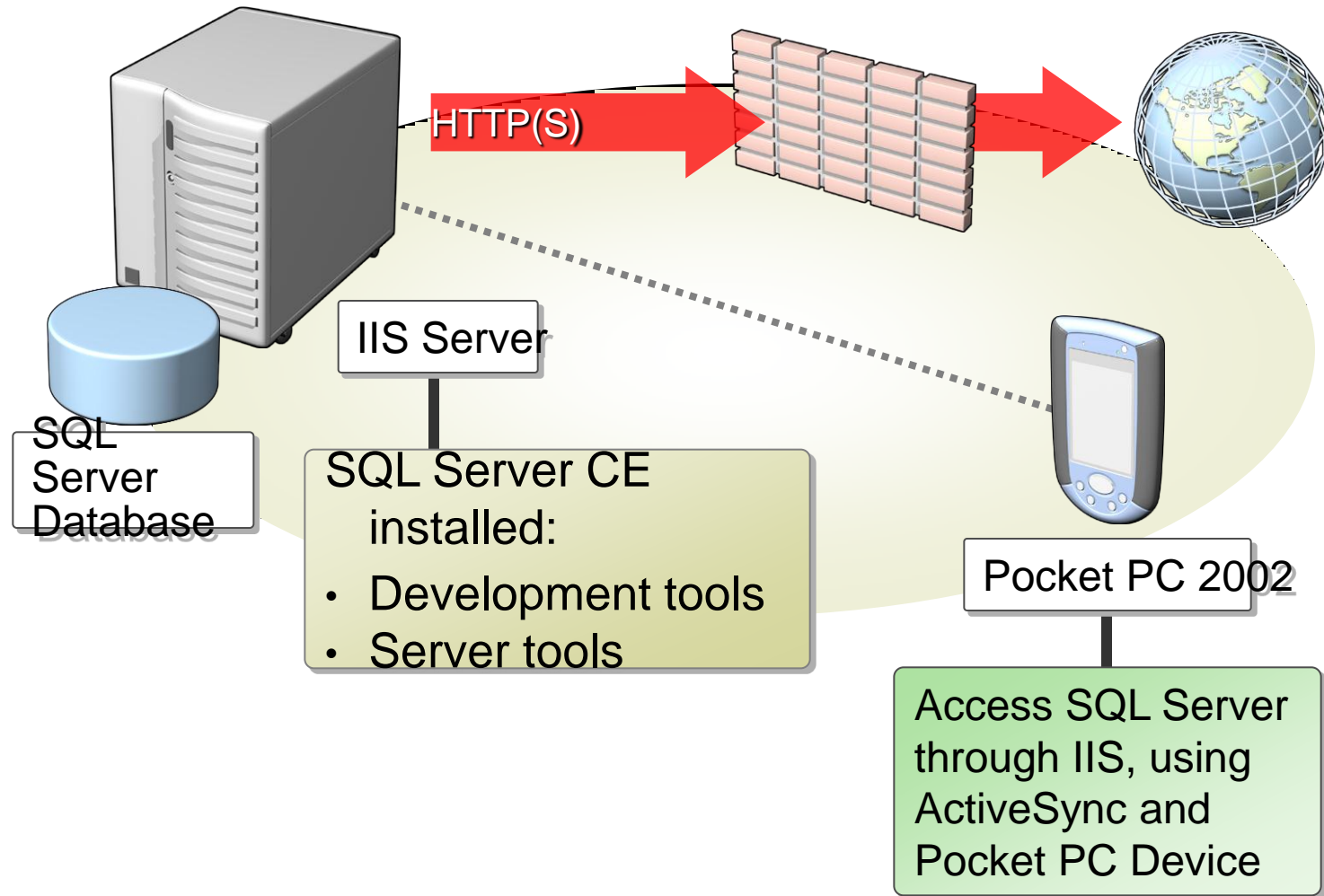
Platform Tools and Technologies

- Common Mobile Deployment Scenarios
- Development Environment: Server
- Development Environment: Tools

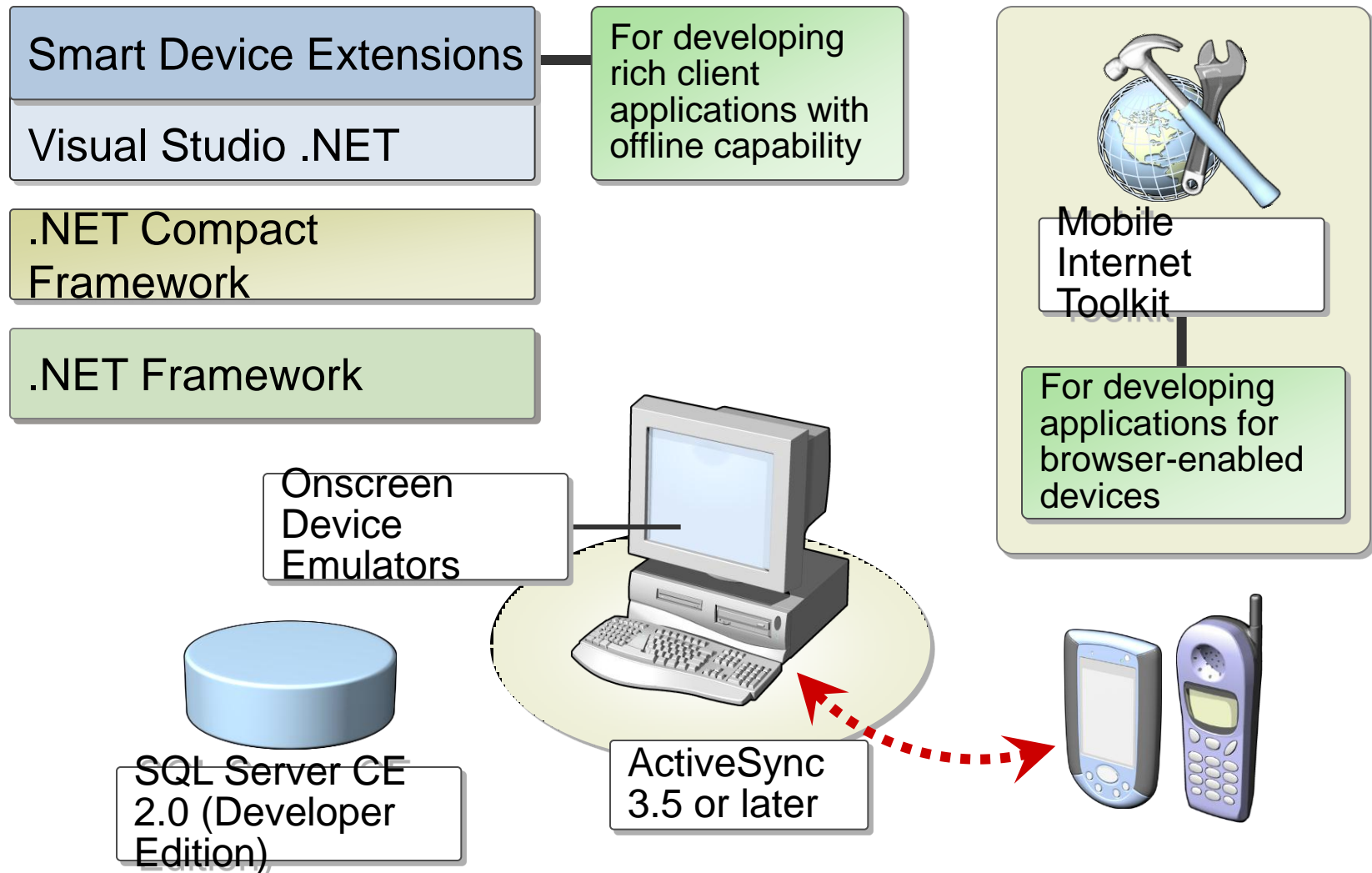
Common Mobile Deployment Scenarios

- Integrating devices with a multitier desktop environment
 - TCP/IP, HTTP, XML, SOAP, and XML Web services
 - Security: user authentication and data encryption
 - Access through firewalls
- Offline vs. online
 - Device detached from the network
 - Data cached locally for use offline
 - Intelligent synchronization of data when connected
 - Wireless connectivity

Development Environment: Server



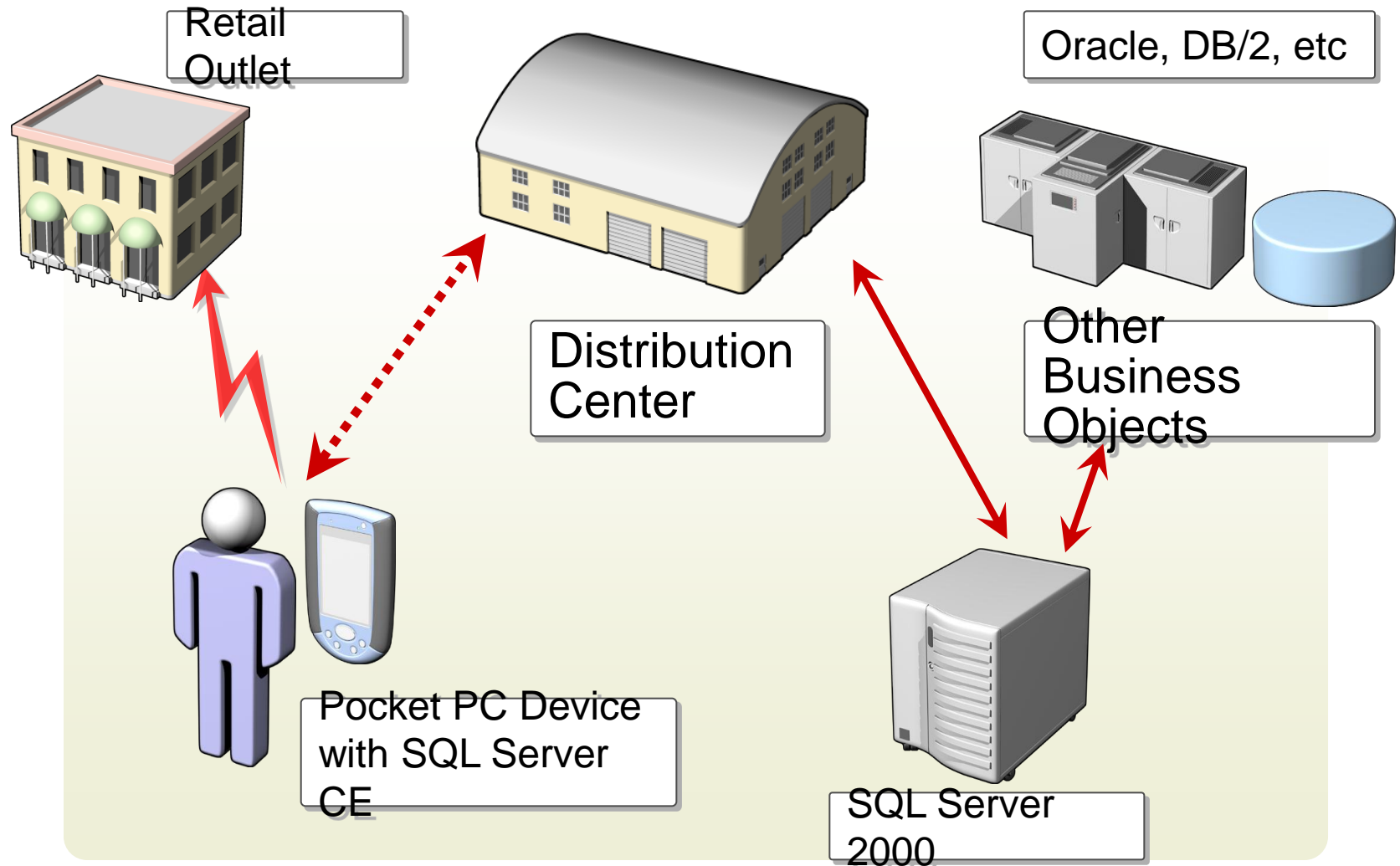
Development Environment: Tools



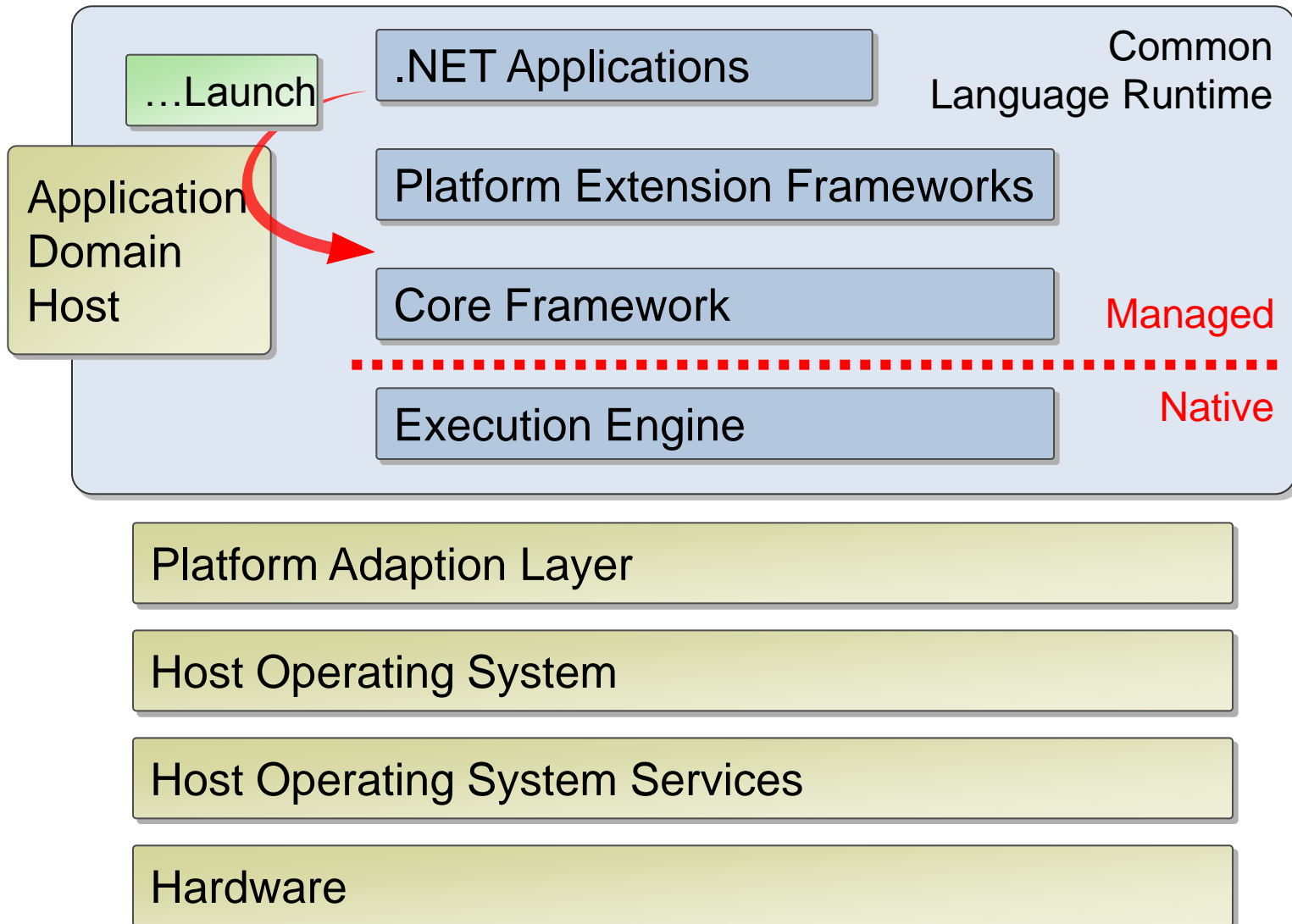
Architectural Design Issues

- Connectivity
 - Wireless
 - Internet
 - Cradle
- Location and work patterns of remote workers
- Synchronization requirements
 - Volume of data to be synchronized
 - Amount and type of change to data
 - Frequency

Sample Mobile Application Scenario



Overview of the .NET Compact Framework



Components Found in a Mobile Application

- Emulators:
 - Mobile applications can be tested and viewed with different emulators some of them are as below
 - **Using your Browser**
 - Since mobile web pages detect your browser, you can test your mobile applications with a standard browser. When a mobile web page detects a web browser it will produce HTML output.
 - **Openwave Simulators**
 - The Openwave mobile browser is the most common browser used in Internet-enabled cellphones. An emulator can be downloaded from <http://www.openwave.com>

Components Found in a Mobile Application

- **The Nokia Mobile Internet Toolkit**
 - This toolkit contains emulators for most of the different Nokia devices. This toolkit can be downloaded from <http://forum.nokia.com>
- **Windows Mobile Development Platform**
 - Here you'll find answers to basic questions regarding the Windows Mobile development platform, along with different downloads:
<http://msdn.microsoft.com/en-us/windowsmobile/bb250560.aspx>

..NET Mobile Components

- .NET Mobile Forms are specialized web forms designed to work on different mobile devices.
- **Mobile Forms**
 - Each mobile page must have at least one mobile form, and each mobile form can have a number of mobile controls. Note that mobile pages can have multiple mobile forms. This is due to the nature of mobile devices. Mobile devices have small screens and it is very normal to navigate between screens with a simple link.
- **Mobile Events**
 - Mobile Controls exposes device independent programmable events.
 - Mobile controls have an object model with programmable properties, methods and events.

.NET Mobile Components (Contd.).

- **NET Mobile Input**

- Input Controls are used to collect input from the mobile user.
- There is a number of mobile controls that support user input.
- The most common input control is perhaps the TextBox control, which was demonstrated in the previous chapter. The TextBox control is perfect for simple user input like names, numbers, identifications and keywords.
- For larger amount of input a TextView control is a better choice. The TextView control allows long multi-line input like the one you need for SMS or other messages

.NET Mobile Components (Contd.).

- **NET Mobile Input Validation**

- Validation controls are used to validate the data entered by a user.
- Validation controls allow you to validate an input control (like a TextBox), and display a message when validation fails.
- Each validation control performs a specific type of validation (like validating against a specific value or a range of values).
- By default, page validation is performed when a command control is clicked. You can prevent validation when a control is clicked by setting the CausesValidation property to false.

- **NET Mobile Lists**

- The Mobile List Control supports different input and display properties.

.NET Mobile Components (Contd.).

- **.NET Mobile Selections**

- The SelectionList Control supports drop down lists, check boxes and radio buttons.

- **.NET Mobile Images**

- Different mobile devices have different display capabilities.
- The Image Control allows the developer to specify different types of images for different types of devices.

- **.NET Mobile Utilities**

- Utility controls support complicated user interfaces with minimum of code.

Create and Deploy Simple Mobile application

References

- Microsoft Developer Network (2012). Retrieved on August 20, 2012, from,
[http://msdn.microsoft.com/en-us/library/ms178619\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms178619(v=vs.85).aspx)
- Learn to create website (1999). Retrieved on August 20, 2012, from, <http://www.w3schools.com/>
- Books
- Microsoft Official Curriculum (MOC) 10953A



Thank You

